OU Academic Integrity Pledge
On my honor I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature: _______________________________ Date: _______________________________

Notes Regarding this Examination

Open Book(s) You may consult any printed textbooks in your immediate possession during the course of this examination.

Open Notes You may consult any printed notes in your immediate possession during the course of this examination.

No Electronic Devices Permitted You may not use any electronic devices during the course of this examination, including but not limited to calculators, computers, and cellular phones. All electronic devices in the student’s possession must be turned off and placed out of sight (for example, in the student’s own pocket or backpack) for the duration of the examination.

Violations Copying another’s work, or possession of electronic computing or communication devices in the testing area, is cheating and grounds for penalties in accordance with school policies.
Part I. Understanding Object-Oriented Design Components & UML

1. (2 points) Which (partial) UML diagram is the most sensible?

2. (2 points) Which (partial) UML diagram is the most sensible?

3. (2 points) Which (partial) UML diagram is the most sensible?

4. (2 points) Which (partial) UML diagram is the most sensible?
Part II. Recognizing Object-Oriented Design Components in Code

Refer to the code listings on Pages 13 and 14 when answering the questions in this part.

5. (2 points) Which line shows an example of declaring an instance variable?
   A. Listing 1, Line 1
   **B. Listing 1, Line 2**
   C. Listing 1, Line 4
   D. Listing 1, Line 5
   E. Listing 1, Line 8

6. (2 points) Which line shows an example of assigning a value to a variable?
   A. Listing 1, Line 1
   **B. Listing 1, Line 2**
   C. Listing 1, Line 4
   D. Listing 1, Line 5
   E. Listing 1, Line 8

7. (2 points) Which line shows an example of instantiating an object?
   A. Listing 1, Line 1
   B. Listing 1, Line 2
   C. Listing 1, Line 4
   **D. Listing 1, Line 5**
   E. Listing 1, Line 8

8. (2 points) Which line shows an example of defining a Class?
   **A. Listing 1, Line 1**
   B. Listing 1, Line 2
   C. Listing 1, Line 4
   D. Listing 1, Line 5
   E. Listing 1, Line 8

9. (2 points) Which line shows an example of a parameter?
   A. Listing 1, Line 1
   B. Listing 1, Line 2
   C. Listing 1, Line 4
   D. Listing 1, Line 5
   **E. Listing 1, Line 8**

10. (2 points) Which lines show an example of overloading?
    **A. Listing 1, Lines 4 and 8**
    B. Listing 1, Lines 14 and 18
    C. Listing 2, Line 5 and Listing 3, Line 11
    D. Listing 2, Lines 5 and 10
    E. Listing 3, Line 4 and Listing 3, Line 7
11. (2 points) Which lines show an example of overriding?
   A. Listing 1, Lines 4 and 8
   B. Listing 1, Lines 14 and 18
   C. **Listing 2, Line 5 and Listing 3, Line 11**
   D. Listing 2, Lines 5 and 10
   E. Listing 3, Line 4 and Listing 3, Line 7

12. (2 points) Which line shows an example of inheritance?
   A. Listing 1, Line 2
   B. Listing 2, Line 5
   C. Listing 2, Line 15
   D. **Listing 3, Line 2**
   E. Listing 3, Line 4

13. (2 points) Which line shows an example of an accessor?
   A. Listing 1, Line 4
   B. **Listing 1, Line 14**
   C. Listing 1, Line 18
   D. Listing 2, Line 15
   E. Listing 3, Line 5

14. (2 points) Which line shows an example of a mutator?
   A. Listing 1, Line 4
   B. Listing 1, Line 14
   C. **Listing 1, Line 18**
   D. Listing 2, Line 5
   E. Listing 2, Line 10

15. (2 points) Which line shows an example of a constructor?
    A. **Listing 1, Line 4**
    B. Listing 1, Line 14
    C. Listing 1, Line 18
    D. Listing 2, Line 5
    E. Listing 2, Line 10

16. (2 points) Which line shows an example of constructor chaining?
    A. Listing 1, Line 22
    B. Listing 2, Line 1
    C. Listing 2, Line 15
    D. **Listing 3, Line 5**
    E. Listing 3, Line 12
17. (2 points) Which line shows an example of a base Class?
   A. Listing 1, Line 1
   B. Listing 1, Line 22
   C. Listing 2, Line 1
   D. Listing 3, Line 2
   E. Listing 3, Line 24

18. (2 points) Which line shows an example of a derived Class?
   A. Listing 1, Line 1
   B. Listing 1, Line 22
   C. Listing 2, Line 1
   D. Listing 3, Line 2
   E. Listing 3, Line 24

19. (2 points) Which line shows an example of Generics?
   A. Listing 1, Line 15
   B. Listing 2, Line 10
   C. Listing 3, Line 2
   D. Listing 3, Line 4
   E. Listing 3, Line 24

20. (2 points) Which line shows an example of the Java Collections Framework?
   A. Listing 1, Line 15
   B. Listing 2, Line 10
   C. Listing 3, Line 2
   D. Listing 3, Line 4
   E. Listing 3, Line 24

21. (2 points) Which of the following would show an example of subclass assignment?
   A. double area = shape.getArea();
   B. Point point = new Point();
   C. double length = Calculator.distance(point1, point2);
   D. Point point = (Point) center;
   E. Shape shape = new Square(corners);

22. (2 points) Which of the following would show an example of dynamic binding?
   A. double area = shape.getArea();
   B. Point point = new Point();
   C. double length = Calculator.distance(point1, point2);
   D. Point point = (Point) center;
   E. Shape shape = new Square(corners);
Part III. Recognizing Object-Oriented Concepts

Refer to the following description when answering the questions in this part.

Nolan, a fantasy novelist and software developer, created the FanDate Class to represent the days, months, seasons, and years of his make-believe world. He made the fields that hold this data private but made numerous public constructor, mutator, and accessor methods with which to interact with objects of Class FanDate, including methods to set and get the months and seasons both by number (taking or returning an int) and by name (taking or returning a String). For example, public int getMonth() would return an int whereas public String getMonth() would return a String.

Later, Nolan created several subclasses of FanDate to represent the way different groups of people within his fantasy world talk about the same dates using different numbering and naming systems for the days, months, seasons, and years. For these Classes, he created mutator and accessor methods with the same names as those in FanDate but with different behavior. For example, in his FanDate Class, public String getMonth() would return “Long Sun” for the eighth month of the year, whereas in his ElfDate subclass, public String getMonth() would return “Hei-viar” instead.

In addition, for his new subclasses he added constructors, mutators, and accessors that take or return FanDate objects, so that he can convert from dates in one subclass to those in another subclass by converting to and from FanDate objects. For example, his ElfDate subclass would have a method public FanDate getDate() that would return a FanDate object that is equivalent to the ElfDate object being queried.

23. (2 points) Having the methods public int getMonth() and public String getMonth() defined in FanDate is an example of which?
   A. encapsulation
   B. overloading
   C. overriding
   D. inheritance
   E. constructor chaining

24. (2 points) Having the methods public void setMonth(int month) and public void setMonth(String month) defined in FanDate is an example of which?
   A. encapsulation
   B. overloading
   C. overriding
   D. inheritance
   E. constructor chaining

25. (2 points) Having the method public String getMonth() defined in FanDate and the method public String getMonth() defined in ElfDate is an example of which?
   A. encapsulation
   B. overloading
   C. overriding
   D. inheritance
   E. constructor chaining
26. (2 points) Having the Classes FanDate and ElfDate is an example of which?
   A. encapsulation  
   B. overloading  
   C. overriding  
   D. inheritance  
   E. constructor chaining

27. (2 points) The choice of access modifiers for FanDate is an example of which?
   A. encapsulation  
   B. overloading  
   C. overriding  
   D. inheritance  
   E. constructor chaining
Part IV. Understanding Object-Oriented Design

Refer to the following description when answering the questions in this part.

Lubna wants to design a software system to keep track of information about her smartphone software apps. Each app has a name, a size (in MB), a version number, a release date, and an icon; was created by a software developer; and can run on one or more smartphone operating systems. Types of apps include games, social apps, productivity apps, utility apps, lifestyle apps, and entertainment apps. Software developers may be individuals or groups, may or may not be businesses, and have names. Operating systems are software with names, version numbers, and release dates.

Data for this system will be stored to and retrieved from files in two formats – a human-readable text format and a machine-readable binary format.

28. (2 points) Which of the following is not an appropriate Class for this software?
   A. App
   B. Comparable
   C. AppNameComparator
   D. OperatingSystem
   E. Driver

29. (2 points) Which Interface should be implemented to allow for objects to be sorted?
   A. Serializable
   B. Cloneable
   C. Comparable
   D. Collections.sort
   E. Collections.binarySearch

30. (2 points) Which relationship should be included in this design?
    A. Developer “is a” Person
    B. App “is a” Software
    C. Software “is a” OperatingSystem
    D. App “is a” Game
    E. App “is a” Utility

31. (2 points) Which relationship should not be included in this design?
    A. App “has a” Name
    B. Developer “has a” App
    C. App “has a” OperatingSystem
    D. OperatingSystem “has a” Name
    E. Developer “has a” VersionNumber
32. (2 points) Which method should not be included in this design?
   A. compareTo
   B. App
   C. getVersionNumber
   D. setName
   **E. hasNext**

33. (2 points) Which field should not be included in App?
   A. productivity
   B. releaseDate
   C. developer
   D. versionNumber
   E. name

34. (2 points) For this software, App should be which of the following?
   A. An Interface
   **B. A Class**
   C. An ArrayList
   D. A method
   E. A field
Part V. Understanding Object-Oriented Design and Java

35. (2 points) Which of the following is a benefit of encapsulation?
   A. Fields are private
   B. Accessors and mutators are public
   C. Data are shielded from accidental modifications
   D. Code size is reduced
   E. Faster execution time

36. (2 points) Which is a key concept behind OO design and programming?
   A. Equations can be represented in code
   B. Software can represent the world
   C. Programs do things
   D. Logical reasoning can prove code correctness
   E. Java is an OO language

37. (2 points) UML Class diagrams show which of the following?
   A. Static relationships between Classes
   B. The flow of execution through OO programs
   C. Objects, Classes, and Interfaces
   D. Class fields and Class method implementations
   E. Class cast exceptions

38. (2 points) Which is a good reason for creating an Interface in Java?
   A. To instantiate objects of the Interface type
   B. To specify a contract that Classes may agree to
   C. To provide encapsulation of data
   D. To reduce code redundancy
   E. All of the above

39. (2 points) Which is a good reason for creating an Abstract Class in Java?
   A. To instantiate objects of the Abstract Class type
   B. To specify a contract that Classes may agree to
   C. To provide encapsulation of data
   D. To reduce code redundancy
   E. All of the above

40. (2 points) Which is a good reason for creating a (concrete) Class in Java?
   A. To instantiate objects of the (concrete) Class type
   B. To specify a contract that Classes may agree to
   C. To provide encapsulation of data
   D. To reduce code redundancy
   E. All of the above
41. (2 points) Which is a good reason for using Generics in Java?
   A. To move errors from run time to compile time
   B. To reduce the amount of casting needed
   C. To cut back on type checking using conditional statements
   D. To allow the same methods to operate on data of different types
   E. All of the above

42. (2 points) Which is an advantage of ArrayList over LinkedList?
   A. ArrayList requires contiguous memory locations for the array
   B. LinkedList can use non-contiguous memory locations
   C. ArrayList allows for binary search
   D. ArrayList doubles in size when it becomes full
   E. ArrayList is a subclass of List

43. (2 points) Which is an advantage of LinkedList over ArrayList?
   A. ArrayList requires contiguous memory locations for the array
   B. LinkedList can use non-contiguous memory locations the list nodes
   C. ArrayList allows for binary search
   D. ArrayList doubles in size when it becomes full
   E. ArrayList is a subclass of List

44. (2 points) Which is a characteristic of HashSet?
   A. HashSet requires contiguous memory locations for the hash table
   B. HashSet has faster lookups for large sets than TreeSet
   C. HashSet cannot contain duplicates
   D. All of the above
   E. None of the above

45. (2 points) Which distinguishes subclasses of Map from subclasses of Collection?
   A. Map subclasses are not indexed; Collection subclasses are
   B. Map subclasses cannot be sorted; Collection subclasses can
   C. Map subclasses use keys to locate values; Collection subclasses use Compare/CompareTo
   D. All of the above
   E. None of the above

46. (2 points) Which is a characteristic of Map?
   A. Map requires contiguous memory for the key/value map
   B. Map has faster lookups for many elements than List
   C. Map cannot contain duplicate keys
   D. All of the above
   E. None of the above
47. (2 points) Which is an advantage of Vector over ArrayList?
   A. Vector is found in legacy code
   B. Vector can use non-contiguous memory locations for the list nodes
   C. Vector allows for binary search
   D. Vector doubles in size when it becomes full
   **E. Vector is synchronized for thread safety**

48. (2 points) Which is the purpose of serialVersionUID?
   A. To identify the version of Java running the code
   **B. To identify the version of the object to be read/written**
   C. To identify the user requesting service
   D. All of the above
   E. None of the above

49. (2 points) Which is an advantage of FileInputStream over FileReader?
   A. FileInputStream is more efficient for non-character data
   B. FileInputStream is more efficient for character data
   C. FileInputStream reads one line at a time
   D. FileReader needs to use BufferedReader
   E. FileInputStream needs to use BufferedInputStream

50. (2 points) Which is an advantage of wrapping a FileReader with a BufferedReader?
   A. FileReader is intended for reading characters from a file; BufferedReader can read from streams as well
   B. FileReader can only read one character at a time; BufferedReader can read multiple characters with one read
   C. BufferedReader can read an entire object with one call to readObject; FileReader cannot
   **D. BufferedReader can improve efficiency by loading more data than currently requested**
   E. Trick question – there is no advantage
Use these listings to answer the questions in Part II. (Feel free to tear off this page for easy reference.)

Listing 1.

```java
public abstract class Shape {
    private Point center;

    public Shape() {
        this.center = new Point();
    }

    public Shape(Point center) {
        this.center = new Point(center);
    }

    public Point getCenter() {
        return new Point(this.center);
    }

    public abstract double getArea();
}
```

Listing 2.

```java
public interface Regular {
    /**
     * @return True if all sides match; False otherwise.
     */
    public abstract boolean sidesMatch();

    /**
     * @return True if all angles match; False otherwise.
     */
    public abstract boolean anglesMatch();

    /**
     * @param length The new length for the sides.
     */
    public abstract void resize(double length);
}
```
Listing 3.

```java
import java.util.ArrayList;

public class Square extends Rectangle implements Regular {
    public Square(ArrayList<Point> corners) throws IllegalArgumentException {
        super(corners);
        if (!sidesMatch(corners)) {
            throw new IllegalArgumentException();
        }
    }

    public boolean sidesMatch() {
        double sideLength = Calculator.distance(getCorner(3), getCorner(0));
        for (int i = 0; i < 3; i++) {
            if (sideLength !=
                Calculator.distance(getCorner(i), getCorner(i + 1))) {
                return false;
            }
        }
        return true;
    }

    public double getArea() {
        double sideLength = Calculator.distance(getCorner(0), getCorner(1));
        return sideLength * sideLength;
    }
}
```